Journal of Computer Science

Original Research Paper

Random Forests for Online Intrusion Detection in Computer Networks

[1]Heitor Scalco Neto, [2]Wilian Soares Lacerda, [1]Rafael Verão Françozo

[1]Instituto Federal de Mato Grosso do Sul, Corumbá-MS, Brazil
[2]Universidade Federal de Lavras, Lavras-MG, Brazil

Article history
Received: 20-04-2021
Revised: 23-06-2021
Accepted: 08-09-2021

Corresponding Author:
Heitor Scalco Neto, Instituto Federal de Mato Grosso do Sul, Corumbá-MS, Brazil
Email: heitorscalco@hotmail.com

Abstract: This study proposes a methodology to build an Online Network Intrusion Detection System by using the Computational Intelligence technique called Random Forests and an API to preprocess the network packets. The experiments were carried out from two network traffic databases: The ISCX (i); and a test database (ii) created with the proposed API in our own network environment. The results obtained with the Random Forests technique show accuracy rates around 98%, bringing significant advances in the area of Intrusion Detection and affirming the high efficiency of the use of the technique to solve problems of intrusion detection in real network environments.

Keywords: Intrusion Detection Systems, Computer Networks, Computational Intelligence, Random Forests

Introduction

The volume of internet traffic in Brazil increased from 0.5 Tb in 2014 to 10.5 Tb in 2021. This growth is due the new services available through the internet, such as banking applications, multimedia and real-time finance systems (Hai, 2021). To accommodate this volume of data reliability and availability, the implementation of tools such as Intrusion Detection Systems (IDS) (Shah, 2017) is necessary to help the network administrators in the security monitoring of the available infrastructure.

Network Intrusion Detection Systems (NIDS) are fundamentally important to ensure information reliability, integrity and availability in a network computer (Moustafa and Slay, 2016). Therefore, this study describes a methodology to develop an online NIDS using random forests (Wu and Banzhaf, 2010; Resende and Drummond, 2018)

The random forests technique is applied and assessed to define the method efficacy of the intrusion detection in a computational environment. In this study, an Application Programming Interface (API) was developed for the proposed NIDS to operate in a real environment. The objective of the API is to capture the network traffic and preprocess the data packet so they can be interpreted by the random forest technique.

The developed API can perform experiments with various network infrastructure simulations and in a real environment. The training of the technique was performed with the ISCX network traffic database. The ISCX is composed of many types of traffic (e.g., VoIP, SSH, HTTP, HTTPS, FTP, etc.).

From the developed API, an auxiliary database was created for a test to address alternative traffic types to those found on ISCX in a smaller scale network, with various operational systems. This can test the effectiveness of the method conducted on different infrastructures and modes of use.

The main contributions of this study are: (i) Validation of the Random Forests technique for the computer network intrusion detection problem; (ii) development of an API for packet capture, pre-processing and integration with various Computational Intelligence techniques; (iii) use of software and/or host independent features.

The results indicate and average score around 98% with ISCX database and Random Forest technique and 96% with the testing database. The principal findings obtained suggests the feasibility of using the Random Forests technique to solve intrusion recognition problems in computer networks.

The following section presents a brief literature review and concepts about network intrusion detection systems and random forests. In the section materials and methods, the methodological procedures and resources used are outlined. The results are presented and discussed in section results. Finally, the last section contains the conclusions of this study.

Theatrical Foundation

This section presents a brief literature review about Network Intrusion Detection Systems and Random Forests.

Science Publications

Network Intrusion Detection Systems

The term "Intrusion Detection" is defined as the process in monitoring events that are occurring in a computational system or network computers, looking for intrusive traffic.

The causes of security incidents have many causes include the propagation of a malware; attackers trying to elevate privileges to access unauthorized systems; and denial of services attacks (Scarfone, 2007). Intrusion Detection Systems are a useful technology for network administrators. The function of IDS is to recognize an anomalous behavior or an intrusive action and report it to the network administrator to take necessary measures (Scalco Neto, 2021).

IDS recognize signature-based or anomaly-based intrusive actions. Signature-based detection systems identify attacks by analyzing previously configured signatures about the standard behavior of some type of attack. In the case of anomaly-based IDS, a network or host pattern is analyzed and traffic is classified as anomalous or normal. The main advantage of an anomaly-based IDS is the possibility of detecting unknown attacks, which is not the case with signature-based detection (Wang, 2009).

A NIDS exclusively analyzes the network traffic without using host-specific information (e.g., memory usage, processing, interfaces), by usually using an interface in promiscuous mode, working as a sniffer (Uchoa, 2009). This type of system usually operates with one or more sensors on the network and a monitoring station. When a sensor detects abnormal activity on the network, an alert is transmitted to the monitoring station that will notify the network administrator about the situation. Figure 1 illustrates an example of an IDS placement on the network (Scalco Neto, 2021), where the IDS is installed on a shared network segment, usually between the public network router and the firewall.

Computational Intelligence techniques, such as Random Forests (Johnson and Jain, 2016; Aburomman and Reaz, 2017), described in the next section, are a method for intrusion recognition in computer networks (Ahmad *et al*., 2021).

Random Forests

To present the Random Forests technique, the concept of Decision Trees must first be explained. Decision Trees use the divide and conquer strategy, whereby a complex problem is broken down into several simpler subproblems in a recursive way. The process of building a Decision Tree is accomplished by selecting an attribute that will be a divisor of the data set (Oshiro, 2013).

Figure 2 shows the structure of a Decision Tree to perform a computer network traffic classification where the result can be "Normal" or "Intrusion". At the root of the tree is "Attribute 88". Hypothetically, if the value of "Attribute 88" >= 1, then the algorithm moves to the left side of the tree to reach "Attribute 5", which performs the same comparison until it reaches a leaf node with an associated class. A Decision Tree can be used to classify samples other than those used for training.

The Random Forests technique uses a set of Decision Trees to constitute its classification "forest". Decision Trees are excellent predictors, but do not always achieve a high generalization ability. On the other hand, the Random Forests technique has excellent accuracy characteristics and high generalizability. This technique uses a recursive partitioning algorithm that combines predictions made by a set of Decision Trees (Biau and Scornet, 2016).

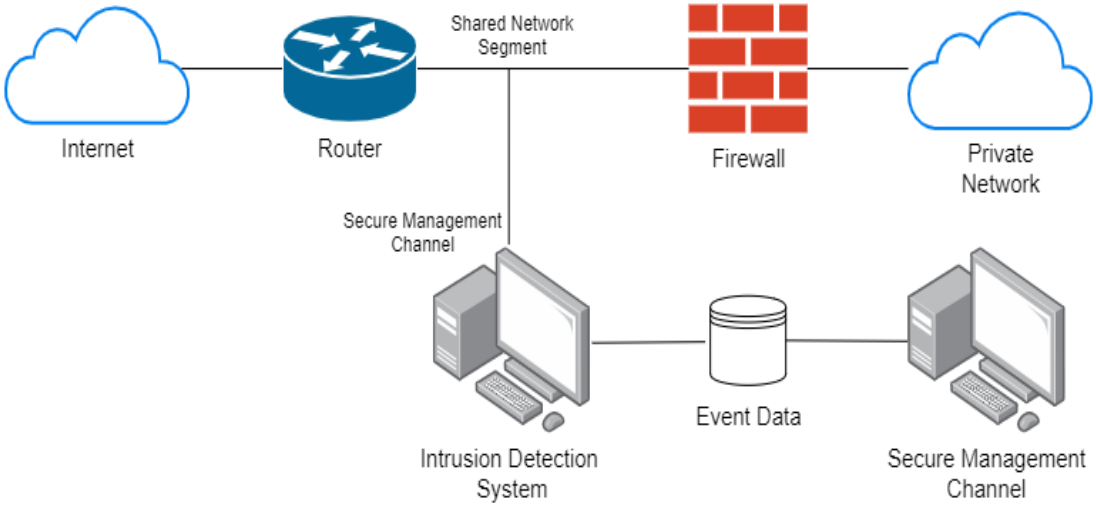

Fig. 1: IDS placement on the network

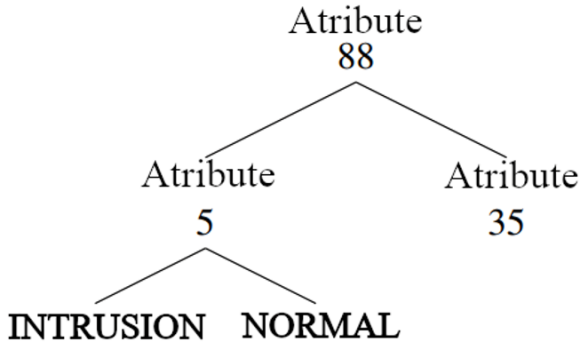

Fig. 2: Example of a decision tree structure

The learning process, or training, of this technique works by creating hundreds (or even thousands) of Decision Trees. Each tree is created based on a small set of samples of the data (or just one sample). Trees are created without an upper limit on the depth of nodes (except when using "pruning" – a process in which the growth of the tree is limited) and forests are formed by aggregating these trees (Boldt, 2014).

The training of Random Forests requires the definition of some parameters. Thus, the algorithm for training Random Forests, using the Scikit-Learn, allows the following parameters to be defined:

- *max_depth*: Maximum depth of the tree
- *max_features*: Maximum number of features used
- *n_estimators*: Maximum number of trees in the forest
- *min_samples_split*: Minimum number of samples to create a node
- *min_samples_leaf*: Minimum number of samples to create a leaf

By performing tests, taking into account the availability of acceptable training time for this type of application, the parameters can be better adjusted, considering limitations of the depth of the trees and other aspects, thus avoiding overfitting (a situation in which the algorithm specializes in the training data, impairing its generalization ability) (Shiravi *et al*., 2012).

Materials and Methods

Performing network traffic analysis and classification, online and in real environment, is always a challenge. According to (Scalco Neto, 2021). The difficulty in building an efficient NIDS is to make the number of true positives large but keep the number of false positives small or even zero. In addition to this challenge, the large amount of information that can be extracted from a network connection can make detection difficult, because most of the time, an intrusion can be characterized by one or more variations of the characteristics of a connection.

To search for a solution to this problem, the Random Forests technique and a set of characteristics extracted from a connection were used. The Random Forests technique presents an advanced degree of generalization and can benefit the results of the classification of intrusive traffic in computer networks. Furthermore, after training the technique, the classification algorithm works with simple operations, making the system more efficient.

In this study, the Scikit-Learn library was used to implement the Random Forests technique. It starts from a principle in which the forest is allowed to grow according to the characteristics of the data and, subsequently, size limitations are performed with processes known as Pruning. The process of solving the proposed problem is presented in detail in this section.

ISCX 2012 Database

The ISCX 2012 database (Shiravi *et al*., 2012) offers some deficiencies found in previously databases (e.g., CAIDA, DARPA and KDD). Therefore, this database provides the logs of connections (normal or intrusive) and all captured traffic (without removing payloads) in a real environment. In addition to the network packet replay capability, the database counts the traffic captured for one full week, totaling 2,450,324 connections. During the traffic capture, several protocols were used, such as: FTP, HTTP, HTTPS, DNS, Netbios, POP3, SMTP, SNMP, SSH and others.

Four scenarios were created and reproduced separately to represent the attacks: Consisting of exploiting vulnerabilities in applications (Exploits); Denial of Service (DoS); Distributed Denial of Service (DDoS); and Brute Force (Shiravi *et al*., 2012).

To apply the database to Computational Intelligence techniques, the arrangement of class data (normal or intrusion) must be evaluated. Preliminary results showed a high degree of unbalance of this database, so the oversampling method was used (He and Garcia, 2009) to equalize the data from the majority class to the minority class.

Packet and Connection Capture and Handling API

To enable the extraction of packet characteristics from a network, as presented in the ISCX 2012 database (Shiravi *et al*., 2012), an application was developed that captures packets, performs pre-processing and sends the information to a classification engine (Random Forests technique). In this way, an Open-Source API is proposed, with the objective of enabling the pre-processing of existing databases, the creation of new databases and online intrusion detection. The API were made available (Scalco Neto, 2021) to the community, with the goal of allowing other authors to generate their own databases in a simplified way and with the ability to change the parameters and the extraction of characteristics from the packets at any time.

According to Salem *et al*. (2014), an effective methodology for pre-processing a database of network packets to detect attacks, especially denial of service attacks, is to use connection vectors (or flows) instead of analyzing individual packets. Thus, it is possible to create a representation of the traffic of a time window (connection) and analyze it as a set. Thus, the main purpose of the API is to capture packets from the network (with the definition of a filter) and process them, so that several "connections" are formed from various packets.

The concept of a "connection" is defined by a data stream, identified by the Unique_id variable. Thus, the Unique_id for TCP and UDP protocols is composed of protocol, source IP address, source port, destination IP address and destination port. In the case of the ICMP protocol, which does not contain ports in its header, the source port and destination port fields have been replaced by the ICMP ID (if present in the header, otherwise it is filled in with -1). Some examples of Unique_id formatting are:

- TCP-177.105.60.1:5800-177.60.20.30:80.
- ICMP-177.60.23.32-177.105.60.1;1200.
- ICMP-177.60.23.32-177.105.60.1; -1.

The selection of characteristics used, extracted from the database by the API for training the Random Forests technique, was based on the proposals of Moustafa and Slay (2016) and Araujo *et al*. (2013). The set of connection characteristics, used in the connection vectors, can be divided into three categories: Characteristics obtained from a connection (Table 1); Characteristics obtained from a Buffer of connection times in a 2 sec past (Table 2); and Characteristics obtained from a Buffer of the last 100 connections (Table 3).

Figure 3 shows the functioning of the API. First, the information of the received packets is extracted and added to the corresponding connection (Fig. 3a). Second, after several iterations, when the connection lifetime is reached, the information is sent to a classification engine, which generates or not the log about the attack information (Fig. 3b).

Finally, the format of the application's output data, which is made available to the Computational Intelligence techniques, via socket, is presented following the model of the following topics, which represent the connection vectors:

- 63.0,TCP, HTTP,S0,280,[...],93.0,100.0,6.0, normal
- 0.0,TCP, SSH,S0,[...],31,5.0,100.0,0.0,100.0, attack

The different modes of operation – Real Environment, Database Construction and Database Read – allow researchers to perform a variety of experiments without having to change the API. If a new mode of operation is required, new features can be added to the API.

The next section presents the process of building a new test database using the developed API.

Process of Building the Test Database

The ISCX 2012 database already provides data from a computer network in a real environment. However, a new database was built with a different infrastructure layout (Computers, Routers, Switches, Smartphones) than the one presented by ISCX 2012. This allows experiments to be conducted to prove that the method is effective for different types of infrastructure and data, not only those provided in the training.

The network infrastructure is definitely an important point for the creation of a database. It is necessary to be cautious about several factors, such as: Topology, services, security and availability of the network assets. The infrastructure used included 8 computers, 1 used to execute the API, 1 to virtualize the servers and 6 others to generate network traffic, as well as 2 Smartphones to use the applications. We configured the mirroring of the switch ports (Mirroring) so that all traffic could be captured (represented by the "Mirror" in Fig. 4). Figure 4 represents a sketch of the network environment used to create the database.

The Switch, Access Point and Router (Fig. 4) were used without any security settings, which ensures the free transmission of network packets (malicious or not). The MAC addresses of the computers and Smartphones were registered in the DHCP server so that the IP addresses did not change during the database creation process.

To represent both normal and anomalous usage of a computer network, several everyday services were executed during the database construction (Table 4). Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks were also executed with TCP (using several flags), UDP and ICMP protocols, as well as port scanning. In the end, 115,030 connections were captured, a sufficient amount to perform effectiveness and generalization tests on the proposed method.

Data Preprocessing and Normalization

The process of normalization and preprocessing are undoubtedly very laborious and important steps to ensure the effectiveness of training the Random Forests technique. Since the API output data format has some non-numerical characteristics, a simple pre-processing is necessary to transform them into numerical inputs.

Because of this, some data entries undergo modifications, for example: The protocols (TCP, UDP and ICMP), connection status Flags (Handshake, Established, Termination, Closed, [...]) and service names (HTTP, HTTPS, POP3, SMTP, SNMP, [...]). Since the classification of various protocols, Flags and/or services are not a quantity that distance values can be defined between them, it is necessary to implement an algorithm that adds a different entry for each protocol. Thus, the protocol entry of the current data is filled out with 1, while the other protocol entries are filled with the value 0, such that:

- HTTP = [1 0 0 0 ... n]
- SMTP = [0 1 0 0 ... n]
- POP3 = [0 0 1 0 ... n]
- HTTPS = [0 0 0 1 ... n]

with *n* the number of distinct Protocols/Flags/Services found in the data.

In this way, the API output data, which used to contain 28 input variables, now have 96 input variables, due to the large number of services accounted for and other information that still needs to be shared in inputs (Protocols and Flags). Finally, the TAG that displays whether the connection is intrusive or not is preprocessed to 0 (Normal) and 1 (Intrusion).

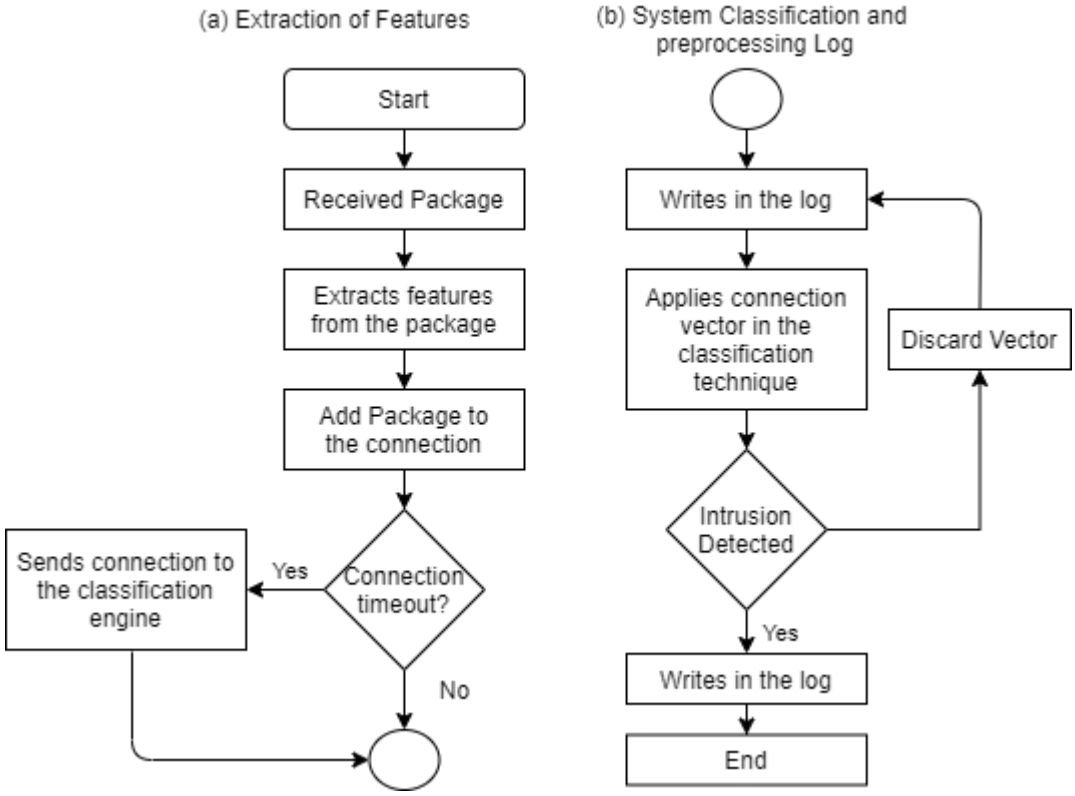

Fig. 3: API Flowcharts

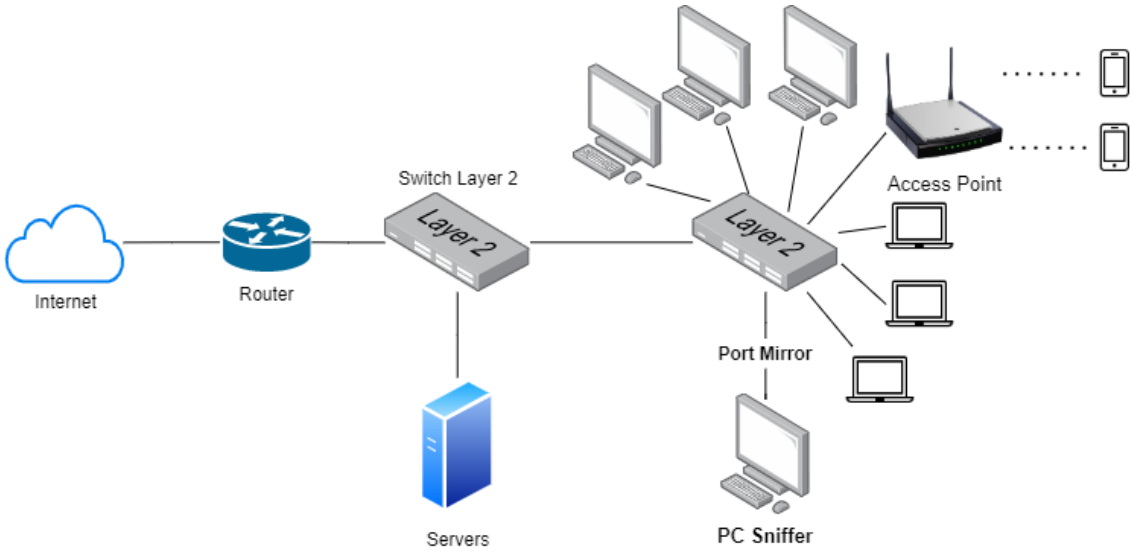

Fig. 4: Network Environment

Table 1: Connection features

#	Characteristic	Description	Type
1	*Duration*	Connection time (in seconds)	continuous
2	*Protocol*	Protocol (Ex.: TCP, UDP, ICMP)	discreet
3	*Service*	Service used (determined by the port)	discreet
4	*Connection Flag*	Connection Status (Ex.: Handshake)	discreet
5	*SourceToDest*	Bytes sent from Source to Destination	continuous
6	*DestToSource*	Bytes sent from Destination to Origin	continuous
7	*Land*	1 if the connection is to/from the same destination/port, 0 the inverse	discreet
8	*Wrong*	Packages with checksum error	continuous
9	*Urgent*	TCP packets with Flag Urgent	continuous
10	*STTL*	TTL of the first package from Origin	continuous
11	*DTTL*	TTL of Destination's first package	continuous
12	*SourceToDestPkts*	Packets sent from Source to Destination	continuous
13	*DestToSourcePkts*	Packets sent from Destination to the Origin	continuous

Table 2: Time buffer features (Standard: 2s)

#	Characteristic	Description	Type
14	*CountSameHost*	Connections to the same host	continuous
15	*CountSameService*	Connections with the same service	continuous
16	*Serror_rate*	% for the same host, with SYN errors (TCP)	continuous
17	*Srv_serror_rate*	% for the same service, with SYN errors (TCP)	continuous
18	*Same_srv_rate*	% for the same service	continuous
19	*Diff_srv_rate*	% for different services	continuous
20	*Srv_diff_host_rate*	% for the same service with different host	continuous

Table 3: Connection buffer features (100 connections)

#	Characteristic	Description	Type
21	*Count*	Connections to the same host	continuous
22	*Srv_count*	Connections with the same service	continuous
23	*Same_srv_rate*	% for the same host, with the same service	continuous
24	*Diff_srv_rate*	% for the same host, with different services	continuous
25	*Same_src_port_rate*	% with the same source port	continuous
26	*Srv_diff_host_rate*	% p/the same service, with different host	continuous
27	*Serror_rate*	% p/the same host, with SYN error (TCP)	continuous
28	*Srv_serror_rate*	% of connections for the same service, with SYN error (TCP)	continuous

Table 4: Services used to build the database

Tool	Protocol
Navigator Web (Google Chrome and Mozilla Firefox)	HTTP, HTTPS
Streaming Audio (Spotify)	UDP and TCP
Hangouts, Skype and Facebook Call	VOIP (UDP and TCP)
FTP Client (Filezilla)	FTP
Email Client (Outlook, Thunderbird, Gmail (Android))	POP3, IMAP, SMTP
Network Monitoring Tool (Cacti)	SNMP
SSH Client (Shell Linux, Putty, WinSCP)	SSH
Server DNS (Bind9) and DHCP (ISC DHCP)	DNS and DHCP
Time Synchronization with Server	NTP
Microsoft Network Discovery	NETBIOS
MEGA and Dropbox Client	TCP, HTTP, HTTPS

Random Forest Technique Training and Analysis

The training step of the Random Forests technique was performed using the Scikit-Learn library. First, the pre-processed training data was mixed and randomly separated, defining 80% of the database for training and 20% for testing.

To ensure consistency of the results, because the initialization of some parameters of the training algorithm was random, the experiments were repeated 10 times. In all repetitions, the data were mixed and separated randomly, maintaining the proportion of 80 and 20%. Besides the analysis of the hit rates, true positives, true negatives, false positives and false

negatives, we obtained the importance of the features used for traffic classification. The Kappa coefficient was also calculated, which is responsible for determining the degree of reliability of the results obtained (Araujo *et al*., 2013).

Results

From the experiments performed with the methodology proposed in this study, using the ISCX 2012 database (Shiravi *et al*., 2012) and the test database, created from the API, we were able to obtain several results using the Random Forests technique, which are presented in detail in this section.

Table 5 presents the hit percentages with the test data from the two databases. The standard deviation of the results is close to zero, which demonstrates a high degree of consistency. Remember that the test database, created from the developed API, was used only as test data, without the data interfering in the training. Thus, the generalization capacity of the method used could be evaluated. Table 5 observes that the Random Forests technique obtained generalization with connections coming from different infrastructures, traffic and applications (Tests database - API).

The definition of the training parameters was carried out empirically, until the expected results were obtained. The depth and maximum number of trees in the forest were limited to avoid overfitting. The limitation of the number of features used was done because, no significant difference in the results occurred when all the features in the database were used. The parameters used for training are:

- Maximum depth: 3
- Maximum features: 25
- Number of trees in the forest: 1000
- Minimum number of samples to create a node: 2
- Minimum number of samples to create a leaf node: 5

Table 6 and 7 present the confusion matrices for the results of the ISCX 2012 database and the developed API database, respectively. The ISCX 2012 database (Table 6) achieves a balance between the numbers of false positives and false negatives, which occurs to a lesser degree in the API database (Table 7). The worst situation in a NIDS is the occurrence of False Negatives because, in this case, intrusive traffic passes through the network undetected. Note that the occurrence of False Negatives was close to zero in both cases. The acronyms TP, TN, FP and FN stand for True Positive, True Negative, False Positive and False Negative, respectively.

According to (Araujo *et al*., 2013), Kappa coefficient values above 0.75 represent a perfect classifier, in which the results presented are consistent and not obtained by mere chance. Figure 5 presents the Kappa coefficient values for both cases (ISCX database and API database), taking into account the classification structure of the first Random Forest training.

Figure 6 shows the important analysis of the features used in the first training of the technique. The remaining values have been excluded from the graph, as they do not present significant importance (very close to zero or equal to zero). The abbreviations “BC” and “BT” signify the origin of the features, which are Connection Buffer (Table 3) and Time Buffer (Table 2), respectively. The high importance of the Count (BC) characteristic is justified by virtue of the ability to detect denial of service attacks by analyzing the volume of traffic to the same host.

Finally, the results obtained with the experiments performed can be compared to other similar proposals found in the literature, which also use the Random Forests technique to classify computer network traffic. Table 8 indicates that the technique proposed here achieved a higher hit than the other proposals using the ISCX 2012 database. The hit rate of the proposed technique using the proprietary database extracted from a real environment and assembled with the developed API is also presented.

Table 5: Percentage of hits using the ISCX 2012 database with random forests

Training	Hit Rate with test data (ISCX 2012 database)	Hit Rate with test data (Developed API database)	Duration
1	98.71%	96.08%	6h14 min
2	98.75%	96.36%	6h15 min
3	98.72%	95.44%	6h17 min
4	98.73%	95.93%	6h12 min
5	98.75%	96.19%	6h26 min
6	98.79%	96.34%	6h17 min
7	98.69%	95.67%	6h20 min
8	98.85%	96.71%	6h11 min
9	98.70%	96.26%	6h22 min
10	98.76%	96.41%	6h06 min
Mean	98.74%	96.14%	-
Standard Deviation	0.07913	0.37442	-

Table 6: Confusion matrix of NIDS with ISCX 2012 database (first training) with random forests

	Positive	Negative	Total
Positive	394.702 (49.02%) – TP	5.749 (0.71%) – FN	400.451
Negative	4.587 (0.56%) – FP	400.126 (49.69%) – TN	404.713
Total	399.289– TP+FP	405.875 – FN+TN	805.164

Table 7: Confusion matrix of NIDS with API database (first training) with random forests

	Positive	Negative	Total
Positive	62,429 (54.27%) – TP	696 (0.60%) – FN	63,125
Negative	3,802 (3.30%) – FP	48,103 (41.81%) – TN	51,905
Total	66,231 – TP+FP	48,799 – FN+TN	115,030

Table 8: Percentage of hits with related work that uses random forests

Proposal	ISCX 2012 database	Own database
This Proposal	98.7%	96.1%
Thaseen *et al*., 2013	96.8%	-
Zhang *et al*., 2008	94.7%	-
Panda *et al*., 2011	80.6%	-

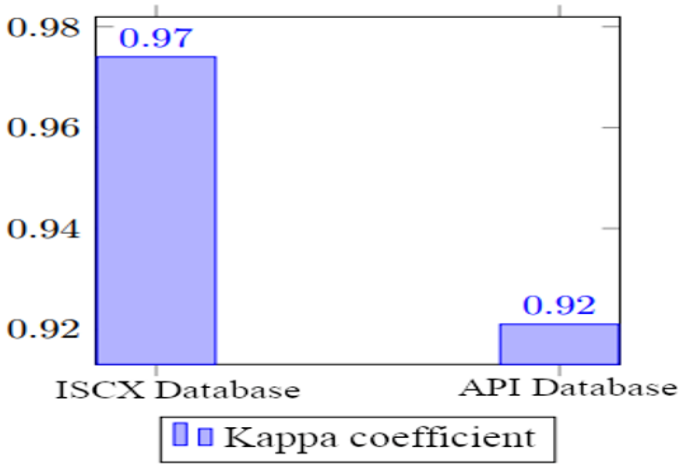

Fig. 5: Kappa coefficient result values

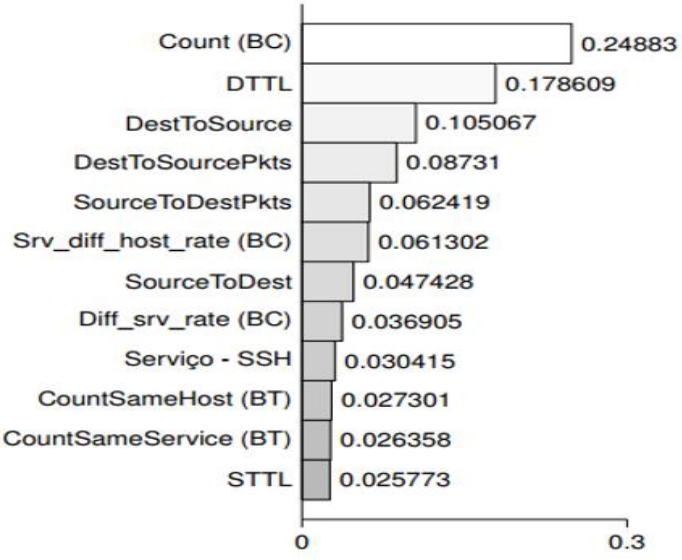

Fig. 6: Importance of the main features of the ISCX 2012 Data-base, with random forests (First Training)

Conclusion

The main objective of this study was to explore, evaluate, present and validate the use of the Random Forests technique to solve intrusion recognition problems in computer networks. The results obtained showed a significant gain in classification accuracy with the proposed methods in relation to other proposals presented in the literature, demonstrating effectiveness in detecting intrusions in various network infrastructures.

The API developed proved to be a great ally to the scientific community, since it allows the creation of new network traffic databases and the pre-processing of existing databases. In this way, a range is opened for other researchers to extract characteristics from network connections and apply them to the desired classification methods.

Continuity proposals included situations like the application of the method in a large-scale network (e.g.,: University, Internet Providers), the insertion of the proposed system in a network equipment (e.g.,: Managed Switch) and the transcription of the API to a low-level language, aiming at increasing the efficiency of the method.

Acknowledgement

This study was developed in a partnership between the Federal University of Lavras (UFLA), located in the state of Minas Gerais, Brazil and the Federal Institute of Mato Grosso do Sul (IFMS), campus of Corumbá, located in the state of Mato Grosso do Sul, Brazil. The authors are grateful for the scholarship provided by UFLA and the time provided by IFMS.

Author's Contributions

Heitor Scalco Neto: Conception, implementation of algorithms and data acquisition, analysis and interpretation of data, drafting the article

Wilian Soares Lacerda: General design, analysis and interpretation of data, reviewing

Rafael Verão Françozo: Analysis and interpretation of data, drafting the article

Ethics

The authors confirm that this article is original and contains unpublished work. All authors have read and approved the manuscript and there are no ethical issues involved.

References

Aburomman, A. A., & Reaz, M. B. I. (2017). A survey of intrusion detection systems based on ensemble and hybrid classifiers. Computers & Security, 65, 135-152. doi.org/10.1016/j.cose.2016.11.004

Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32 (1), 1-29. doi.org/10.1002/ett.4150

Araujo, N. V. D. S. (2013). Kappa-PSO-ARTMAP Fuzzy: uma metodologia para detecção de intrusos baseado em seleção de atributos e otimização de parâmetros numa rede neural ARTMAP Fuzzy https://repositorio.unesp.br/handle/11449/100302

Biau, G., & Scornet, E. (2016). A random forest guided tour. Test, 25(2), 197-227. doi.org/10.1007/s11749-016-0481-7

Boldt, A. S. (2014). Coleções nucleares e associação do teor de óleo de cártamo com variáveis ecogeográficas por inteligência computacional. https://www.locus.ufv.br/handle/123456789/1400

Hai, T. H., Khiem, N. T., & Phuclk, N. H. (2021). Toward an Online DoS/DDoS Classification: An Empirical Study for Network Intrusion Detection Systems. Journal of Computer Science, 17 (3), 304-318. doi.org/10.3844/jcssp.2021.304.318.

He, H., & Garcia, E. A. (2009). Learninga from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9), 1263-1284. doi.org/10.1109/TKDE.2008.239

Johnson, S. R., & Jain, A. (2016). An Improved Intrusion Detection System using Random Forest and Random Projection. International Journal of Scientific & Engineering Research, 7 (10), 1424-1430.

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective, 25(1-3), 18-31. doi.org/10.1080/19393555.2015.1125974.

Oshiro, T. M. (2013). Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica (Doctoral dissertation, Universidade de São Paulo).https://www.teses.usp.br/teses/disponiveis/95/95131/tde-15102013-183234/publico/monografia

Panda, M., Abraham, A., Das, S., & Patra, M. R. (2011). Network intrusion detection system: A machine learning approach. Intelligent Decision Technologies, 5 (4), 347–356. doi.org/10.3233/IDT-2011-0117

Resende, P. A. A., & Drummond, A. C. (2018). A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys (CSUR),

51(3), 1-36. doi.org/10.1145/3178582

Salem, M., Reißmann, S., & Buehler, U. (2014, February). Persistent dataset generation using real-time operative framework. In 2014 International Conference on Computing, Networking and Communications (ICNC) (pp. 1023-1027). IEEE. doi.org/10.1109/ICCNC.2014.6785478

Scalco Neto, H. (2021). NIDS Project. https://github.com/heitorscalco/NIDSProject/

Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). NIST special publication, 800(2007), 94. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-94.pdf

Shah, J. (2017, January). Understanding and study of intrusion detection systems for various networks and domains. In 2017 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-6). IEEE. doi.org/10.1109/ICCCI.2017.8117726

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security, 31(3), 357-374. doi.org/10.1016/j.cose.2011.12.012

Thaseen, S., & Kumar, C. A. (2013, February). An analysis of supervised tree based classifiers for intrusion detection system. In 2013 international conference on pattern recognition, informatics and mobile engineering (pp. 294-299). IEEE. doi.org/10.1109/ICPRIME.2013.6496489

Uchoa, J. Q. (2009). Algoritmos Imunoinspirados aplicados em segurança computacional: utilizaçao de algoritmos inspirados no sistema imune para detecçao de intrusos em redes de computadores. https://repositorio.ufmg.br/handle/1843/GRFO-7VWGHS

Wang, J., 2009. Computer network security: theory and practice. Springer, Massachusetts. ISBN: 978-3-540-79697-8, pp: 400.

Wu, S. X., & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. Applied soft computing, 10(1), 1-35. doi.org/10.1016/j.asoc.2009.06.019

Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(5), 649-659. doi.org/10.1109/TSMCC.2008.923876